Defocus Video Matting

Defocus Video Matting

Morgan McGuire *
Brown University

Wojciech Matusik
MERL

Hanspeter Pfister
MERL

John F. Hughes
Brown University

Frédo Durand
MIT

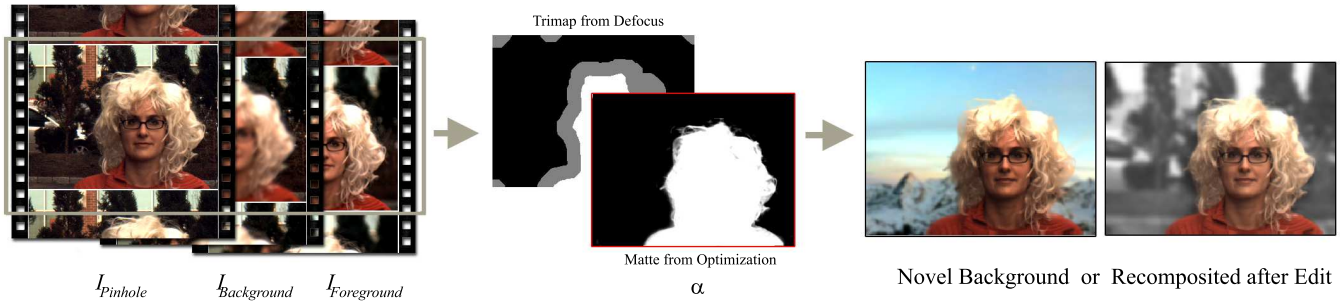

Figure 1: *From three pixel-aligned video streams with varying focus we automatically extract a trimap. We then solve for the most probable matte by constrained optimization and post-process to remove noise from ill-conditioned pixels. Defocus matting pulls mattes with sub-pixel detail from natural images without user intervention and in situations where blue screen matting is impractical.*

Abstract

Video matting is the process of pulling a high-quality alpha matte and foreground from a video sequence. Current techniques require either a known background (e.g., a blue screen) or extensive user interaction (e.g., to specify known foreground and background elements). The matting problem is generally under-constrained, since not enough information has been collected at capture time. We propose a novel, fully autonomous method for pulling a matte using multiple synchronized video streams that share a point of view but differ in their plane of focus. The solution is obtained by directly minimizing the error in filter-based image formation equations, which are over-constrained by our rich data stream. Our system solves the fully dynamic video matting problem without user assistance: both the foreground and background may be high frequency and have dynamic content, the foreground may resemble the background, and the scene is lit by natural (as opposed to polarized or collimated) illumination.

1 Introduction

Matting and compositing are some of the most important operations in image editing, 3D photography, and film production. Matting or "pulling a matte" refers to separating a foreground element from an image by estimating a color F and opacity α for each foreground pixel. Compositing is used to blend the extracted foreground element into a new scene. α measures the coverage of the foreground object at each pixel, due to either partial spatial coverage or partial temporal coverage (motion blur). The set of all α values is called the alpha matte or the alpha channel.

Because of its importance, the history of matting is long and colorful [Smith and Blinn 1996]. The original matting approaches require a background with known, constant color, which is referred to as *blue screen matting*, even though green is preferred when shooting with digital cameras. Blue screen matting has been perfected for half a century and is still the predominant technique in the film industry. However, it is rarely available to home users, and even production houses would prefer a lower-cost and less intrusive alternative. On the other end of the spectrum, rotoscoping [Fleischer 1917] permits non-intrusive matting but involves painstaking manual labor to draw the matte boundary on many frames.

Ideally, one would like to pull a high-quality matte from an image or video with an arbitrary (unknown) background, a process known as *natural image matting*. Recently there has been substantial progress in this area [Ruzon and Tomasi 2000; Hillman et al. 2001; Chuang et al. 2001; Chuang et al. 2002; Sun et al. 2004]. Unfortunately, all of these methods require substantial manual intervention, which becomes prohibitive for long video sequences and for non-professional users.

The difficulty arises because matting from a single image is fundamentally under-constrained [Smith and Blinn 1996]. The matting problem considers the input image as the *composite* of a foreground layer F and a background layer B, combined using linear blending [Porter and Duff 1984] of radiance values for a pinhole camera:

$$I_P[x,y] = \alpha F + (1-\alpha)B, \tag{1}$$

where αF is the (pre-multiplied) image of the foreground element against a black background, and B is the image of the (opaque) background in the absence of the foreground. Matting is the inverse problem with seven unknowns $(\alpha, F_r, F_g, F_b, B_r, B_g, B_b)$ but only three constraints (I_{Pr}, I_{Pg}, I_{Pb}). Note that blue screen matting is easier to solve because the background color B is known.

We advocate a *data-rich imaging* approach to video matting. We introduce a novel imaging setup that records additional information during capture, thereby constraining the original ill-posed problem. This preserves the flexibility of non-intrusive techniques since only the imaging device is modified, not the scene, while offering full

*Email: morgan@cs.brown.edu

automation. The additional information we exploit comes from *defocus*: three pixel-aligned video streams are recorded with different focusing distance and depth of field. We call our approach *defocus matting* and demonstrate how it can pull high-quality mattes from video sequences without user assistance.

The novel contributions in this paper are the application of a multiparameter camera to video matting; an automatic method to extract crude, conservative mattes called *trimaps*; and a novel optimization method for defocus matting from multiparameter video.

2 Related Work

In practice, blue-screen matting requires more than a uniform-color screen– the screen must be carefully lit to avoid shadows, maintain uniform intensity, and avoid reflecting blue light on the subject. We seek to avoid or at least reduce the need to control the background and illumination. The natural world with its $1/f$ noise distribution tends to provide texture for us.

Most natural image matting approaches [Chuang et al. 2001; Hillman et al. 2001; Chuang et al. 2002; Rother et al. 2004] require user-defined trimaps to compute the color distributions of F and B in known regions. Using these distributions they estimate the most likely values of F and B for the unknown pixels and use them to solve the matting equation (1). Bayesian matting [Chuang et al. 2001] and its extension to video [Chuang et al. 2002] arguably produce the best results in many cases. But user-painted trimaps are needed at keyframes; this becomes tedious for long video sequences. We propose a solution that operates without user assistance and that can be applied as a batch process for long video clips today and will eventually run in real-time on live video as it is recorded.

In addition, robust estimation of color distributions works only if F and B are sufficiently different in the neighborhood of an unknown pixel. Our technique can pull a matte where foreground and background have similar color distributions if there exists sufficient texture to distinguish them in defocused images. In cases where there are neither high frequencies nor color differences, the natural image matting problem is insoluble. Where the defocus problem is ill-conditioned we introduce regularization terms inspired by previous matting algorithms to guide our solver to a physically probable solution.

Poisson matting [Sun et al. 2004] solves a Poisson equation for the matte by assuming that the foreground and background are slowly varying compared to the matte. Their algorithm interacts closely with the user by beginning from a hand-painted trimap and offering painting tools to correct errors in the matte. Defocus matting works best with high-frequency backgrounds, so it complements Bayesian and Poisson matting, which are intended for low-frequency backgrounds.

The basic strategy of acquiring more pixel-aligned image data has been successfully used in other computer graphics and computer vision applications, such as high-dynamic range [Debevec and Malik 1997; Nayar and Branzoi 2003], confocal [Levoy et al. 2004], super-resolution [Ben-Ezra and Nayar 2004], depth estimation from defocus [Nayar et al. 1996], depth estimation from focus [Pentland 1987; Asada et al. 1998]. The focus-based depth estimation techniques inspired our approach. However, they are not directly applicable to the natural video matting problem. First, reconstruction techniques cannot produce fractional alpha values, so they are limited to opaque super-pixel structures. Second, depth-from-focus requires hundreds of defocussed images for a single frame so it is only appropriate for stills. Third, depth-from-defocus is only reliable with active illumination, and for natural matting we desire a passive technique that does not interfere with the scene. Infrared illumination avoids visible patterns but does not work many scenes– radar, active techniques work best with diffusely reflective surfaces and can give poor results on mirror reflective or absorptive (black) surfaces that do not reflect the active illumination towards the camera, or in the presence of IR interference, e.g., from direct sunlight. These drawbacks apply to non-focus active IR systems like the Zcam [Yahav and Iddan 2002].

Zitnick et al. [2004] were the first to create a passive, unassisted natural video matting system. They capture video on a horizontal row of eight sensors spaced over about two meters. They compute depth from stereo disparity using sophisticated region processing, and then construct a trimap from depth discrepancies. The actual matting is computed by the Bayesian matting [Chuang et al. 2001] algorithm on a single view; Zitnick et al.'s contributions are the physical system and stereo trimap extraction.

Our system also uses multiple sensors, but they share an optical axis using beam splitters. This avoids view dependence problems associated with stereo sensors (e.g., reflections, specular highlights, occlusions) and allows for an overall smaller camera. We pull our trimap using defocus and use information from all of our cameras during matting. In this paper, we concentrate on the matting problem instead of the trimap problem. Our trimap region processing is simple compared to that of Zitnick et al. A hybrid trimap method combining our depth-from-defocus with their depth-from-stereo and region processing would likely be superior to either alone.

The closest work to ours is a scene reconstruction method by Favaro and Soatto [2003]. Both methods use defocussed images and both use gradient descent minimization of sum-squared error, a common framework in both graphics and vision. They solved for coarse depth and binary alpha; we solve for alpha only but achieve sub-pixel results and an orders of magnitude speedup (precisely, O(image size)) by using exact differentiation. We work with color video instead of monochrome still images, which necessitates a new capture system and calibration. Color is needed to over-constrain the problem and video to reconstruct partly occluded background pixels. We extend Favaro and Soatto's regularization terms; because matting equations can be ill-conditioned at some pixels, finding good regularization terms continues to be an active area of research, e.g., [Apostoloff and Fitzgibbon 2004; Blake et al. 2004].

Schechner et al. [Schechner et al. 2000] were the first to use a depth-from-focus system to recover overlapping objects with fractional alpha. They drive a motorized CCD axially behind the lens to capture hundreds of images with slightly varying points of focus. Depth is recovered by selecting the image plane location that gave the best focussed image (this is similar to how autofocus works in commercial cameras). This method is limited to static scenes and requires very high frequencies everywhere. In contrast, our optimizer is driven towards a correct solution even in low-frequency areas by the regularization terms.

3 Overview

We over-constrain the matting problem by capturing multiple synchronized video streams using a multi-sensor camera. Beam splitters allow all sensors to share a virtual optical center yet have varying parameters. The *pinhole* sensor has a small aperture that creates a large depth of field. It is nominally focused on the foreground. The *foreground* and *background* sensors have large apertures, creating narrower depths of field. The foreground sensor produces sharp images for objects within about $\frac{1}{2}$m of depth of the foreground object and defocuses objects farther away. The background sensor produces sharp images for objects from about 5m to infinity and defocuses the foreground object. Because the background camera's depth of field is very large and there is no parallax between our cameras, a background with widely varying depths can still be well approximated as a plane for the purpose of matting.

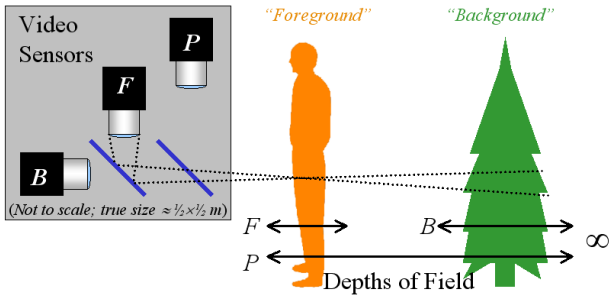

Figure 2: *The multiparameter camera captures video streams that share an optical center but vary in focus, exposure, and aperture.*

Given these three streams, we pose matting as an optimization problem. For each frame, we express the optical formation of the three input images I_F, I_P, I_B as a function of unknown images α, F and B using a model of the defocus blur. Our optimizer seeks to minimize an error function of the sum-squared differences between the simulated images and the observed ones. For each pixel there are seven unknown "scene" values α, $F_{\{r,g,b\}}$, and $B_{\{r,g,b\}}$ and nine constraint values $I_{P\{r,g,b\}}$, $I_{F\{r,g,b\}}$, and $I_{B\{r,g,b\}}$ from the sensors, so the problem is over-constrained. To speed the optimizer's convergence we automatically create trimaps using depth-from-defocus, and choose initial values for the unknowns that are likely near the true solution. The initial foreground values F_0 are created by automatically painting known foreground colors into the unknown regions. The initial background values B_0 are created by reconstructing occluded areas from neighboring frames and then painting into never-observed areas. The initial coverage values α_0 are computed by solving the pinhole compositing equation using F_0 and B_0. Defocus matting is poorly conditioned if the foreground and background have the same color, if the scene lacks high frequencies, or if the images are under- or over-exposed. To avoid local minima and stabilize the optimizer in these poorly conditioned areas, we add low-magnitude regularization terms to the optimizer.

The challenge in solving defocus matting by optimization is choosing an error function that is efficient to evaluate and easy to differentiate, because the core of an optimizer is this error function, which will be invoked a few hundred times per frame. Our error function is the sum-squared pixel value error between the captured images and composites rendered from the unknowns. Evaluating and differentiating it naively make the probelm intractable. To move towards a global minimum, the optimizer must find the gradient of the error function (i.e., the partial derivatives with respect to each unknown variable). For $320 \times 240 \times 30$fps color video there are over 13 million unknowns per second of video. Numerically evaluating the gradient, for instance, requires invoking the error function once for each variable, and in our case this involves rendering three full-resolution images. Assuming a very fast distributed ray tracer can render the images in three seconds, so that a single call to the error function takes three seconds, it would take *years* to optimize a few seconds of video.

To solve the problem efficiently, we therefore leverage our knowledge of the error function. Symbolically manipulating expressions allows us to avoid numerical computation. We introduce a very fast approximation to the image synthesis problem for specific scenes, which allows us to evaluate the error function in milliseconds. We replace numerical evaluation of the error derivative with a symbolic derivative based on our synthesis equations.

4 Multiparameter Camera

We have built a camera that can directly capture multiparameter video using eight computer vision sensors at the nodes of a binary tree of beam-splitters. Our camera is a reconfigurable device; we can change sensor settings and record, for example, high dynamic range (varying exposure), super resolution (varying sub-pixel offset), and multi-spectral (varying wavelength) video. It is also a portable workbench, containing an optical breadboard for mounting components, a PC, lights, and battery packs for two hours of continuous operation bolted to a $1 \times \frac{1}{2} \times 2$m wheeled cart. For matting we operate only three sensors, which occupy a $\frac{1}{4} \times \frac{1}{4}$m area of the breadboard.

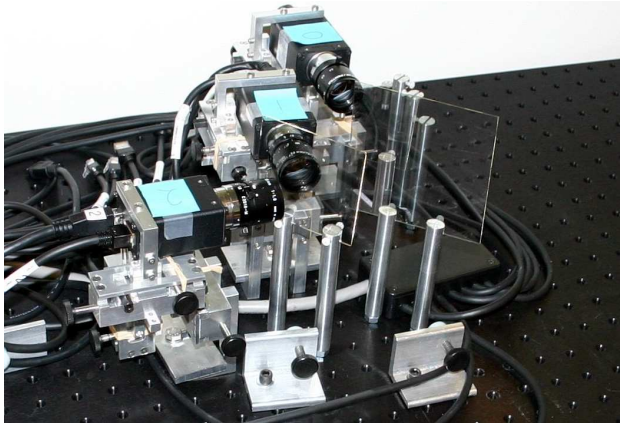

Figure 3: *Multiparameter video camera. An enclosing case (removed for this photo) blocks the beam splitters from ambient light.*

The sensors are Basler a601fc computer vision cameras that produce $640 \times 480 \times 30$ fps of Bayer-encoded video. Unlike many consumer cameras, these sensors respond linearly to incident radiance. We connect each sensor to a separate FireWire bus on a single PC and wire the hardware shutter triggers to the parallel port, which we strobe at 30 fps. We equip each sensor with an $f = 50$mm lens. The pinhole sensor sits immediately behind the first beam splitter. It has an aperture of $f/12$ and is focused on the foreground plane, since a correct matte is more important than a correct reconstructed background. The foreground and background sensors have $f/1.6$ apertures and are behind a second beam splitter. Although they each receive only half the light of the pinhole sensor, their large apertures capture too much illumination so we place neutral density filters in front of the lenses. As long as the image is not under- or over-exposed, the color calibration process corrects remaining intensity differences between sensors.

4.1 Calibration

We calibrate the physical cameras to within a few pixels and then correct remaining error in software. The process is repeated after relocation because our camera mounts are easily disturbed.

The primary challenge is aligning the optical axes so that there is no parallax between cameras. We place a large white poster of a quincunx pattern of five black bull's eyes 8m from the camera. We place a transparent sheet printed with a small version of the pattern 2m from the camera so that it precisely occludes the distant pattern in one camera's view. We translate the remaining cameras until the distant pattern is overlapped by the near one in each view. As with the sights on a rifle, the alignment of points on two planes under projection guarantees no parallax.

Focusing a camera also changes the size of the image produced, so even perfectly aligned cameras produce differing images. We correct for this with an affine transformation in software. We have used two methods with equal success. Before capturing data we shoot a scene containing a bright LED moving perpendicular to the optical axis at the foreground camera's midfield depth. The LED's centroid is easy to track (even when defocused) and provides a set

of points from which we solve for the least-squares transformation (called a homography matrix). We repeat the process for the far plane. When it is inconvenient to use the LED (e.g., the far plane is in the center of a busy street), we can also manually select feature points in a single set of three still frames after capture. Note that the calibration process is performed only when the camera has been physically disrupted—we do not calibrate each video sequence.

We color correct the images by solving a similar problem in color space. Here, the feature points are the colors of an image of a color chart and the affine transformation is a color matrix.

We apply color and position correction in real-time to all streams in OpenGL on a GeForce 6800 GPU for both preview and capture. Each video frame is loaded into a rectangular texture and rendered with the homography as the texture-coordinate transformation matrix and a fragment shader that multiplies each texel value by the color matrix.

5 Defocus Composites

5.1 Notation

We introduce the following notation to compactly express discrete imaging operations. Monochrome images are 2D matrices. Image matrices are multiplied componentwise, not with matrix multiplication, and must have matching dimensions. A *multiparameter image* is sampled across camera parameters like wavelength, focus, and time as well as pixel position. We represent it with a 3D or larger matrix, e.g., $C[x,y,\lambda,z,t]$. Note that although we do not explicitly do so in this paper, this notation and our matting algorithm extend to images with more than three color samples and to other parameters like polarization, sub-pixel position, and exposure.

Expressions like $C[\lambda,z]$, where some parameters are missing, denote a sub-matrix containing elements corresponding to all possible values of the unspecified parameters, i.e., `C(:,:,L,z)` in Matlab. The equations in this paper have generally the same form in x and y, so we frequently omit the y; we also omit the z, λ, and t parameters when they do not vary throughout an equation.

Let the convolution $F \otimes G$ have the same size as F and be computed by extending edge values of F by half the size of G, so that it is well defined near the edges. Let $\mathrm{disk}(r)$ be the function that returns a normalized, discrete 2D disk filter kernel with radius r pixels (i.e. an image of an anti-aliased disk with unit integral). When $r < \frac{1}{2}$, the disk becomes a impulse δ that is one at $[0,0]$ and zero elsewhere. Convolution with an impulse is the identity; convolution with a disk is a blur.

A vector hat denotes a multiparameter image unraveled into a column vector along its dimensions in order, e.g., $\vec{F}[x+W((y-1)+H(\lambda-1))] = F[x,y,\lambda]$ for an image with $W \times H$ pixels and 1-based indexing. To distinguish them from image matrices, elements of unraveled vectors are referenced by subscripts. Linear algebra operations like matrix-vector multiplication, inverse, and transpose operate normally on these vectors.

5.2 Lens Images

Equation 1 is the discrete compositing equation for a pinhole camera. In this section we derive an approximate compositing equation for a lens camera with a non-zero aperture, which forms a pinhole by defocus. In computer graphics, lens cameras are traditionally simulated with distributed ray tracing. We instead use a filter-based approach more common to computer vision, which is well suited to the image-based matting problem.

Defocus occurs because the cone of rays from a point in the scene intersects the image plane at a disk called the *point spread function* (PSF) aka circle of confusion. Figure 4 shows the geometry of the situation giving rise to a PSF with pixel radius

$$r = \frac{f}{2\sigma\#}\left|\frac{z_R(z_F-f)}{z_F(z_R-f)}-1\right|, \tag{2}$$

where the camera is focused at depth z_F, the point is at z_R, # is the aperture (f-number), f is focal length, and σ is the width of a pixel [Glassner 1995]. Depths are positive distances in front of the lens.

A single plane of points perpendicular to the lens axis with pinhole image αF has a defocused lens image given by the convolution $\alpha F \otimes disk(r)$ [Hecht 1998]. Adding the background plane to the scene complicates matters because the background is partly occluded near foreground object borders. Consider the bundle of rays from a partly occluded point to the aperture. The light transport along each ray is modulated by the α value where the ray intersects the foreground plane. Instead of a cone of light reaching the aperture from each background point, a cone cut by the α image reaches the aperture. The PSF therefore varies for each point on the background: it is zero for occluded points, a disk for unoccluded points, and a small cut-out of the α image for partly occluded points. However, we can express it simply in two out of three important cases:

1) Pinhole. When $f\sigma$ is very small or # is very large, r is less than half a pixel at both planes and Equation 1 holds.

2) Focused on Background. When the background is in focus its PSF is an impulse (zero radius disk with finite integral). Rays in a cone from B are still modulated by a disk of $(1-\alpha)$ at the foreground plane, but that disk projects to a single point in the final image. Only the average value, and not the shape, of the α disk intersected affects the final image. The composition equation is:

$$I_B = (\alpha F) \otimes \mathrm{disk}(r_F) + (1 - \alpha \otimes \mathrm{disk}(r_F))B\,. \tag{3}$$

3) Focused on Foreground: When the background is defocused, its PSF varies along the border of the foreground object. Here the correct image expression is complicated[1] and slow to evaluate, so we use the following approximation [Asada et al. 1998]:

$$I_F \approx \alpha F + (1-\alpha)(B \otimes \mathrm{disk}(r_B))\,, \tag{4}$$

which blurs the background slightly at foreground borders.

Let $a[x,y]$ be a 2D matrix, $F[x,y,\lambda]$ and $B[x,y,\lambda]$ be 3D matrices. We generalize the two-plane compositing expression with a function of the scene that varies over two discrete spatial parameters, a discrete wavelength (color channel) parameter λ, and a discrete focus parameter $z \in \{1,2,3\}$:

$$\begin{aligned} &C(\alpha,F,B)[x,y,\lambda,z] = \\ &\qquad (\alpha F[\lambda]) \otimes h[z] + (1 - \alpha \otimes h[z])(B[\lambda] \otimes g[z])\big|_{[x,y]}\,, \end{aligned} \tag{5}$$

where 3D matrices h and g encode the PSFs:

$$h[x,y,z] = \begin{cases} \delta[x,y], & \text{z=1} \\ \mathrm{disk}(r_F)[x,y], & \text{z=2} \\ \delta[x,y], & \text{z=3} \end{cases} \tag{6}$$

$$g[x,y,z] = \begin{cases} \delta[x,y], & \text{z=1} \\ \delta[x,y], & \text{z=2} \\ \mathrm{disk}(r_B)[x,y], & \text{z=3}\,. \end{cases} \tag{7}$$

Constants r_F and r_B are the PSF radii for the foreground and background planes when the camera is focused on the *opposite* plane.

Equation 5 can be evaluated efficiently: for small PSF radii, we can simulate a 320×240 lens camera image in ten milliseconds.

[1] See [Bhasin and Chaudhuri 2001] for the expression and a discussion of the interaction between occlusion and defocus.

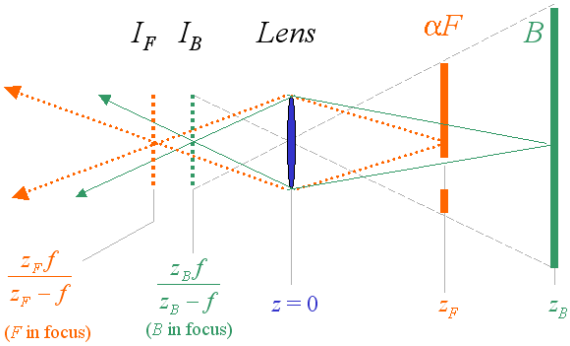

Figure 4: The point spread function is the intersection of a cone and the imager. The cone contains all rays from a point to the aperture.

6 Trimap from Defocus

A *trimap* segments a pinhole image into three mutually exclusive and collectively exhaustive regions expressed as sets of pixels. These sets limit the number of unknowns and steer initial estimates. Hand-painted trimaps are common in previous work; we instead produce them automatically as follows.

Areas in the scene that have high-frequency texture produce high-frequency image content in I_P and exactly one of I_F and I_B. We use this observation to classify pixels with high-frequency neighborhoods into three regions based on the z values for which they appear sharp, as shown in the sample trimap in Figure 5. Sets Ω_B and Ω_F contain pixels that are respectively "definitely background" ($\alpha = 0$) and "definitely foreground" ($\alpha = 1$). Set Ω contains "unknown" pixels that may be foreground, background, or some blend. This is the set over which we solve for the matte.

Many surfaces with uniform macro appearance actually have fine structural elements like the pores and hair on human skin, the grain of wood, and the rough surface of brick. This allows us to detect defocus for many foreground objects even in the absence of strong macro texture. We must use lower thresholds to detect high frequencies in the background, where only macro texture is visible.

We create a first classification of the foreground and background regions by measuring the relative strength of the spatial gradients:

$$\begin{aligned} & \text{Let } D = \text{disk}(\max(r_F, r_B)) \\ \Omega_{F1} &= \text{erode}(\text{close}((|\nabla I_F| > |\nabla I_B|) \otimes D > 0.6, D)), D) \quad (8) \\ \Omega_{B1} &= \text{erode}(\text{close}((|\nabla I_F| < |\nabla I_B|) \otimes D > 0.4, D)), D)\,, (9) \end{aligned}$$

where erode and close are morphological operators [Haralick et al. 1987] used to achieve robustness. The disk should be approximately the size of the PSFs. We then classify the ambiguous locations either in both Ω_{F1} and Ω_{B1} or in neither:

$$\Omega = (\tilde{\Omega}_{F1} \cap \tilde{\Omega}_{B1}) \cup (\Omega_{F1} \cap \Omega_{B1}). \quad (10)$$

Finally, we enforce the mutual exclusion property:

$$\begin{aligned} \Omega_F &= \Omega_{F1} \cap \tilde{\Omega} \quad (11) \\ \Omega_B &= \Omega_{B1} \cap \tilde{\Omega}\,. \quad (12) \end{aligned}$$

7 Minimization

We pose matting as an error minimization problem for a single frame of video and solve for each frame independently.

Assume we know the approximate depths of the foreground and background planes and all camera parameters. These are reasonable

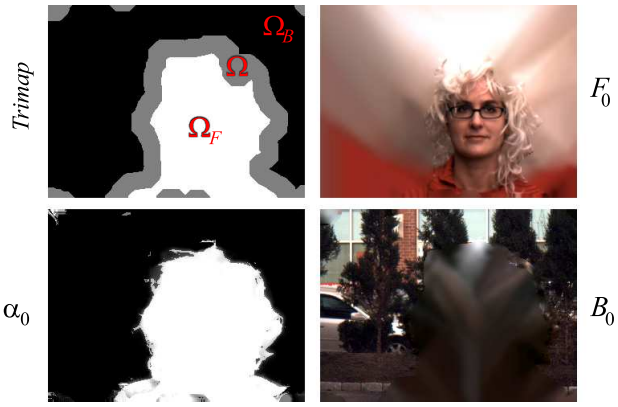

Figure 5: *Clockwise from upper left: Trimap from defocus, foreground "outpainted" to fill space, background reconstructed from adjacent frames and inpainted, and matte via pinhole composition.*

assumptions because digital cameras directly measure their parameters, and from the lens-imager distance we can derive the depths to the planes, if otherwise unknown. The foreground and background need not be perfect planes, just lie within the foreground and background camera depth fields. Because depth of field is related hyperbolically to depth, the background depth field may even stretch to infinity.

Let $u = [\vec{\alpha}^T \vec{B}^T \vec{F}^T]^T$ be the column vector describing the entire scene (i.e., the unknowns in the matting problem) and $\vec{C}(u)$ be the unraveled composition function from Equation 5. The unraveled constraints are $\vec{I} = [\vec{I_P}^T \vec{I_B}^T \vec{I_F}^T]^T$. The solution to the matting problem is a scene $u^\star$ for which the norm of the error vector $\vec{E}(u) = \vec{C}(u) - \vec{I}$ is minimized:

$$\text{Let } Q(u) = \sum_k \frac{1}{2} \vec{E}_k^2(u) \quad (13)$$

$$u^\star = \underset{u}{\text{argmin}}\, Q(u)\,. \quad (14)$$

Note that the scalar-valued function Q is quartic because it contains the terms of the form $(\alpha[x]F[i])^2$.

Iterative solvers appropriate for minimizing such a large system evaluate a given scene u and choose a new scene $u + \Delta u$ as a function of the vector $\vec{E}(u)$ and the Jacobian matrix $\mathbf{J}(u)$. The latter contains the partial derivative of each element of $\vec{E}(u)$ with respect to each element of u,

$$\mathbf{J}_{k,n}(u) = \frac{\partial \vec{E}_k(u)}{\partial u_n}. \quad (15)$$

It may help the reader to think of k as an index into the unraveled constraints and n as an index into the unraveled unknown array. Henceforth, we will generally write $\vec{E}$ rather than $\vec{E}(u)$ and so on for the other functions of u to simplify the notation in the presence of subscripts.

A gradient descent solver moves opposite the gradient of Q:

$$\begin{aligned} \Delta u &= -\nabla Q = -\nabla \sum_k \frac{1}{2} \vec{E}_k^2 \quad (16) \\ \text{so } \Delta u_n &= -\frac{\partial \sum_k \frac{1}{2} \vec{E}_k^2}{\partial u_n} = -\sum_k \left(\vec{E}_k \frac{\partial \vec{E}_k}{\partial u_n} \right) \quad (17) \\ \text{hence } \Delta u &= -\vec{E}^T \mathbf{J}\,. \quad (18) \end{aligned}$$

The gradient descent solver has a space advantage over other methods like Gauss-Newton and Levenberg-Marquardt because it

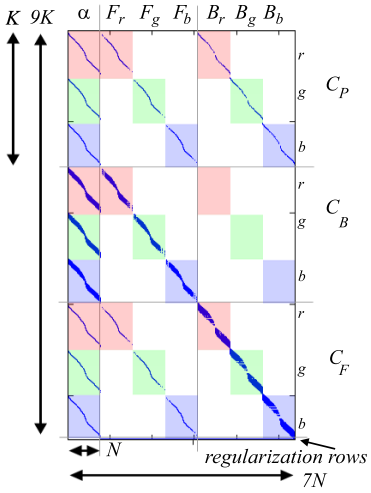

Figure 6: *Sparsity structure of J. Background colors in the diagram represent the color channel corresponding to each term. Dark blue marks non-zero elements.*

never computes the pseudo-inverse of the Jacobian. This is important because the vectors and matrices involved are very large. Let N be the number of unknown pixels and K be the number of constrained pixels; $\text{length}(u) = 7N, \text{length}(\vec{C}(u)) = 9K$, and $\text{size}(\mathbf{J}) = 7N \times 9K$. For 320×240 images, $\mathbf{J}$ has about 6×10^9 elements.

We now derive a simple expression for the elements of the Jacobian and argue that it is sparse, so computing Δu is feasible when we do not need the (non-sparse) inverse of $\mathbf{J}$. By definition, the elements are:

$$\mathbf{J}_{k,n} = \frac{\partial(\vec{C}_k(u) - \vec{I}_k)}{\partial u_n} = \frac{\partial \vec{C}_k(u)}{\partial u_n}. \tag{19}$$

To evaluate this, we expand the convolution from equation 5. We must change variables from packed 1D vectors indexed by k to images indexed by x:

$$C[x,z,\lambda] = \sum_s \alpha[s]F[s,\lambda]h[x-s,z] + \left(1 - \sum_s \alpha[s]h[x-s,z]\right)\sum_s B[s,\lambda]g[x-s,z]\,. \tag{20}$$

Examination of this expansion shows that $\mathbf{J}$ is both sparse and simple. For example, consider the case where unknown u_n corresponds to $F[i,\lambda]$. In a full expansion of equation 20, only one term contains $F[i,\lambda]$, so the partial derivative contains only one term:

$$\frac{\partial C[x,\lambda,z]}{\partial F[i,\lambda]} = \alpha[i]h[x-i,z]\,. \tag{21}$$

The expressions for α and B derivatives are only slightly more complicated, with (potentially) non-zero elements only at:

$$\begin{aligned}\frac{\partial C[x,\lambda,z]}{\partial \alpha[i]} &= h[x-i,z]\left(F[i,\lambda] - \sum_s B[\lambda,s]g[x-s,z]\right) \\ &= h[x-i,z]\left(F[i,\lambda] - (B[\lambda] \otimes g[z])\,[x]\right)\end{aligned} \tag{22}$$

$$\begin{aligned}\frac{\partial C[x,\lambda,z]}{\partial B[i,\lambda]} &= g[x-i,z]\left(1 - \sum_s \alpha[s]h[x-s,z]\right) \\ &= g[x-i,z]\left(1 - (\alpha \otimes h[z])\,[x]\right).\end{aligned} \tag{23}$$

Note that the summations in these last two cases are just elements of convolution terms that appear in $\vec{E}$, so there is no additional cost for computing them. Figure 6 shows the structure of an actual Jacobian. The non-zero elements (blue) are in blocks along diagonals because of the unraveling process. Each column is effectively an unraveled image of several PSFs.

7.1 Trust Region and Weights

The gradient tells us the direction to change u to reduce error. We use a so-called dogleg trust region scheme (see [Nocedal and Wright 1999] for discussion) to choose the magnitude. The idea is to take the largest step that also most decreases error. We begin with a trust region of radius $S = 1$. Let $u' = \max(0, \min(1, u + \frac{S\Delta u}{|\Delta u|}))$. If $|\vec{E}(u')| < |\vec{E}(u)|$, then we assume we have not overshot the minimum and repeatedly double S until the error increases above the lowest level seen this iteration. If $|\vec{E}(u')| > |\vec{E}(u)|$, then we assume we have overshot and take the opposite action, repeatedly halving S until we pass the lowest error seen this iteration. When S becomes very small (e.g., 10^{-10}) or the error norm shrinks by less than 0.1%, we assume we are at the local minimum and terminate the optimization process.

Because our initial estimates are frequently good, we weigh the first N elements of Δu by constant $\beta_\alpha \approx 3$ to influence the optimizer to take larger steps in α. This speeds convergence without shifting the global minimum. We also reduce the magnitude of $\vec{E}$ elements corresponding to I_P by a factor of $\beta_P \approx \frac{1}{4}$. The narrow aperture and long exposure of the pinhole image produce more noise and motion blur than I_F and I_B, and this prevents over-fitting the noise. It also reduces the over-representation in $\vec{E}$ of in-focus pixels that occurs because F and B are in focus in two of the constraint images and defocused in one each.

7.2 Regularization

In foreground areas that are low frequency or visually similar to the background, there are many values of u that will satisfy the constraints. We bias the optimizer towards likely solutions. This is *regularization* of the optimization problem, which corresponds to having a different prior for a maximum likelihood problem. Regularization also helps pull the optimizer out of local minima in the error function and stabilizes the optimizer in areas where the global minimum is in a flat region of many possible solutions.

We extend the error vector $\vec{E}$ with p new entries, each corresponding to the magnitude of a $7N$-component *regularization vector*. Calling these regularization vectors $\varepsilon, \phi, \gamma, \ldots$, the error function Q now has the form:

$$\begin{aligned}Q(u) = \sum_k \vec{E}_k^2 &= \left[\sum_{k=1}^{9K} \vec{E}_k^2\right] + \vec{E}_{9K+1}^2 + \vec{E}_{9K+2}^2 + \ldots \\ &= \sum_{k=1}^{9K} \vec{E}_k^2 + \beta_1 \frac{9K}{7N}\sum_n^{7N} \varepsilon_n^2 + \beta_2 \frac{9K}{7N}\sum_n^{7N} \phi_n^2 + \ldots\,.\end{aligned} \tag{24}$$

Let e be one of the regularization vectors. Each summation over n appears as a new row in $\vec{E}$ and $\mathbf{J}$ for some $k > 9K$:

$$\vec{E}_k = \left(\beta \frac{9K}{7N}\sum_n e_n^2\right)^{\frac{1}{2}} \tag{25}$$

$$\mathbf{J}_{k,n} = \frac{\partial \vec{E}_k}{\partial u_n} = \frac{\beta}{\vec{E}_k}\frac{9K}{7N}\sum_i \left[e_i \frac{\partial e_i}{\partial u_n}\right]. \tag{26}$$

The factor of $\frac{9K}{7N}$ makes the regularization magnitude invariant to the ratio of constraints to unknowns and the scaling factor β allows us to control its significance. We use small weights on the order of $\beta = 0.05$ for each term to avoid shifting the global minimum.

The outer square root in the $\vec{E}$ expression is canceled by the square in the global error function Q. In the computation of $\vec{E}^T\mathbf{J}$, the $\vec{E}_k$ factor in the denominator of Equation 26 cancels; in what follows, we will give, for each regularization vector, both e_n and $(\vec{E}^T\mathbf{J})_n$. We choose regularization vectors that are both easy to differentiate and efficient to evaluate: the summations over i generally contain only one non-zero term.

Straightforward but tedious differentiations lead to the expressions for $(\vec{E}^T\mathbf{J})_n$ in each of the following regularization terms; the details are omitted.

Coherence: spatial gradients are small,

$$e_n = \frac{\partial u_n}{\partial x};\ (\vec{E}^T\mathbf{J})_n = -\frac{\partial^2 u_n}{\partial x^2}\,. \tag{27}$$

We apply separate coherence terms to α, F, and B, for each color channel and for directions x and y. The α gradient constraints are relaxed at edges (large values of $|\nabla I_P|$) in the original image. The F gradient constraints are increased by a factor of ten where $|\nabla\alpha|$ is large. These allow sharp foreground edges and prevent noise in F where it is ill-defined.

Discrimination: α is distributed mostly at 0 and 1,

$$e_n = u_n - u_n^2;(\vec{E}^T\mathbf{J})_n = (u_n - u_n^2)(1-2u_n) \;\Big|\; 1 \le n \le N\,. \tag{28}$$

Background Frequencies should appear in B:

Let $G = I_B - I_F \otimes \mathrm{disk}(r_F)$

$$e_n = \frac{\partial u_n}{\partial x} - \frac{\partial \vec{G}_n}{\partial x};(\vec{E}^T\mathbf{J})_n = -\frac{\partial^2 u_n}{\partial x^2} \;\Big|\; 4N+1 \le n \le 7N\,. \tag{29}$$

8 Results

We convert 640 × 480 Bayer video to RGB using Malvar et al.'s demosaicing [2004]. To further reduce Bayer artifacts and noise, in some cases we average every four pixels to produce a 320 × 240 stream. A Matlab implementation of our method pulls mattes in about seven minutes per frame, comparable to the time for a recent user-assisted matting technique [Sun et al. 2004].

Defocus matting is unassisted and each frame is computed independently, so frames can be processed in parallel. We built a "matting farm" of eight PCs for an amortized processing time under one minute per frame. The method could someday execute in real-time; over 95% of the solution is obtained within thirty seconds of optimization per frame and the hardware assisted trimap computation already produces a real-time preview.

8.1 Ideal Images

We tested our method on still images and video of rendered and real scenes. We begin with the synthetic scenes, which allow us to compare our results to previous work, measure deviation from a known ground truth, and to measure the effect of error sources on the result. Synthetic scenes are constructed from three images, α, F, and B using the filter-based synthesis approximation and reasonable foreground and background depths (e.g., 2m and 6m).

Figure 7 shows a hard "corner case" in which foreground and background are both noise. Although our system can pull a very good trimap for this scene, we intentionally replaced the trimap with one in which the unknown region encompasses the whole image. Despite the lack of a good trimap, defocus matting is still able to pull a good matte. This demonstrates that the quality of the result is independent of the trimap, unlike previous approaches that learn

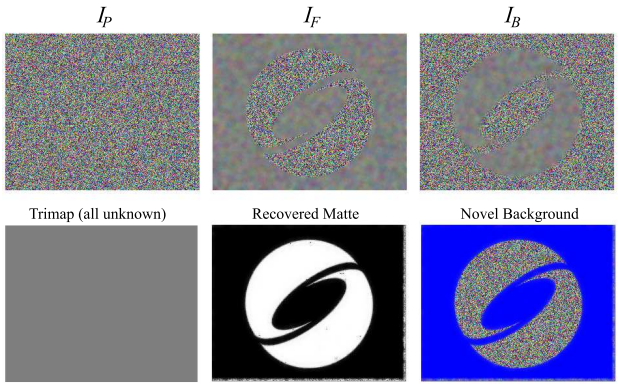

Figure 7: *A synthetic scene where the foreground is the SIGGRAPH logo made from noise and the background is also noise. The top row shows the images seen by the cameras. The bottom row shows a trimap that treats all pixels as unknown and the matting result.*

Figure 8: *Neither foreground (a yellow square) nor background (a green rectangle) has any texture, but we still pull a good matte because of the regularization terms.*

a color model from the trimap. Because there is no trimap, convergence is slow and processing time was 3 hours for 320 × 240 pixels. Because both foreground and background have identical color distributions, this scene is an impossible case for color model approaches. Moreover, in the pinhole image, the foreground and background are indistinguishable! A matting approach working from a that image alone could not succeed because there is no information.

Figure 8 shows a yellow square in front of a green screen. This would be an ideal case for blue screen, Bayesian, and Poisson matting. Given a hand-painted trimap, we recover a perfect matte as well. We cannot, however, pull an automatic trimap for this scene because of the artificial lack of texture. This is the opposite corner case, and demonstrates that the minimization still converges even in the absence of texture, relying mostly on regularization terms.

To compare against scenes used in previous work we use the published recovered α and αF. We reconstruct the occluded part of the background in a paint program to form B. An ideal result in this case is a reconstruction of the matte from the previous work; we cannot exceed their quality because we use their result as ground truth.

Figure 9 uses data that is now standard for matting papers, a difficult case with complex hair structure (data set from [Chuang et al. 2001], ground truth from [Sun et al. 2004]). We pull a good matte in 30 seconds at 320 × 240 and 5 minutes at 640 × 480. The nonlinear performance falloff occurs because the smaller resolution can be solved as two large blocks while the larger requires solving many more subproblems and letting information propagate. Figure 10 shows an enlarged area of the hair detail from figure 9. Our matte reflects the single-pixel hair structures from the ground truth structure. The contrast level is slightly higher in our matte than the ground truth matte because the regularization terms reject the broad $\alpha \approx 0.1$ areas as statistically unlikely.

If we replace the initial estimates in the previous example with noise in the "unknown" region (see figure 9), the optimizer still converges to the correct solution, albeit slower; this matte took three minutes to pull. This shows that results are independent of the quality of the initial estimates.

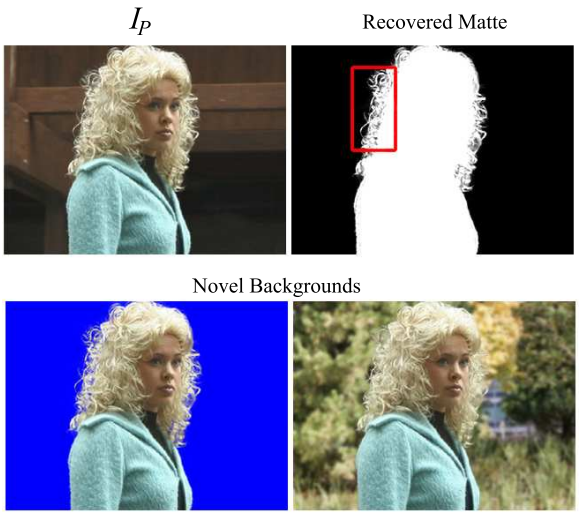

Figure 9: *Pinhole image, recovered matte, and two recomposited images for a challenging case with fine hair structure.*

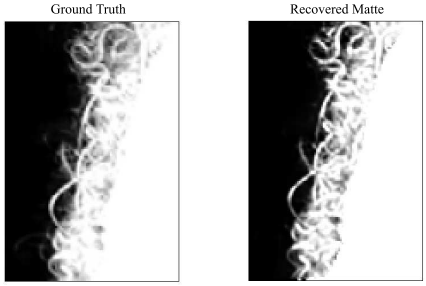

Figure 10: *Comparison of the synthetic ground truth to the recovered value for of the hair from the red box of figure 9.*

8.2 Sources of Error

There are many sources of error in a real multiparameter images. The most significant are translation (parallax) error between sensors, over- and under-exposure, and different exposure times (i.e., amounts of motion blur) between cameras. These lead to pixels in different sensors that are not close to the values predicted by the composition equation. Other sources of error have less impact. Our approach appears to be robust to color calibration error, which does not affect the gross intensity magnitudes and yields the same least-squares solution. Radial distortion by the lenses is small compared to the translation errors observed in practice due to miscalibration. Our gradient constraints and the inherently wide filter support of defocus tend to minimize the impact of sensor noise. Figure 12 shows a case where the inputs have been shifted a few pixels, one has been rotated about 2 degrees, and the colors of two have been shifted; remarkably, the algorithm still converges, although the result is not very satisfactory.

8.3 Real Images

In real images, the sources of error discussed in the previous section produce low-order noise near the boundaries of the trimap, as holes in the matte, and as opaque regions that should have been transparent in the matte. This can be seen in figure 14, where a light-colored building panel blends into over-exposed hair in the foreground during one frame of a video sequence. On the left side there is also small amount of noise, at the border of Ω_B. This is correctable; we overcome small amounts of noise by modulating the recovered matte towards black within 3 pixels of Ω_B and removing disconnected components, i.e., tiny, bright α regions that do not touch the

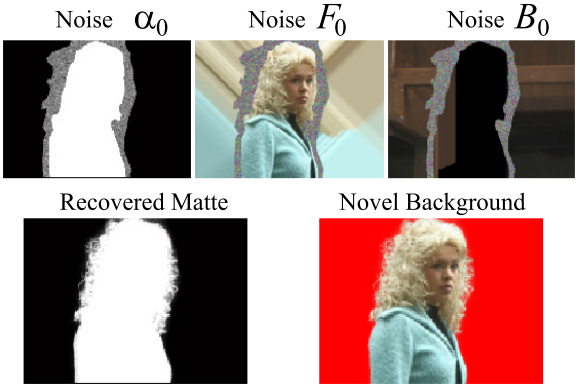

Figure 11: *Even in the absence of good initial values (the foreground and background in the "uncertain" region are seeded with noise) we are able to pull a good matte.*

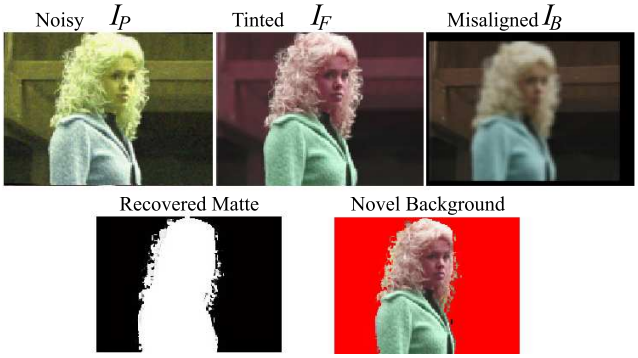

Figure 12: *Robustness against rotation, translation, and hue-shift.*

region enclosing Ω_F. We likewise fix holes well within objects by filling the region 12 pixels inside the $\alpha < 0.1$ isocurve. The matte results in this paper are the unprocessed optimization result unless otherwise noted; in the video they are all post-processed. When the sources of error are small so that the problem is well-posed, defocus matting succeeds at pulling good mattes for real images and video.

Figure 15 again shows an example of matting with hair, this time from a real image. The algorithm pulls a good matte, reconstructing both the fine hair structure and the motion blur on the left side of the face. This scene has no high frequencies in the background so we hand-drew a trimap for the first frame. To pull a matte for video in these circumstances, we experimented with propagating the trimap forward, so that areas well within the $\alpha = 0$ and $\alpha = 1$ regions (at least 10 pixels in) of the final matte for frame i become Ω_B and Ω_F for frame $i+1$. This works well on the hair, but cannot account for objects moving in from off-camera; it is also prone to propagate any matte errors forward. The lower part of the figure shows three frames of the alpha-matte at different points in the video sequence.

We observed "checkerboard" artifacts in the matte under two circumstances. We originally used a fast Bayer interpolation that produced a half-pixel shift in the red and blue channels. Because the beam-splitters mirror the image left-to-right, this shift was inverted between cameras and led to inconsistent alpha values in a checkered pattern. Better Bayer interpolation removes this artifact. A

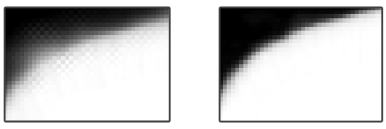

Figure 13: *Ringing in the matte when the coherence terms are too large; correct results with small terms.*

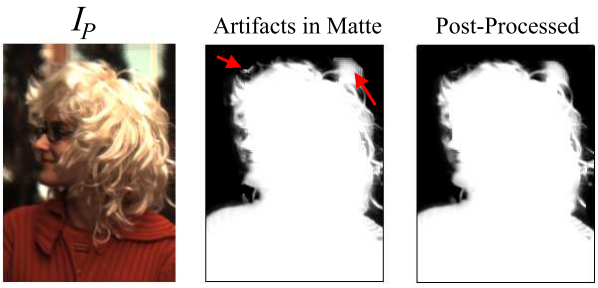

Figure 14: *Noise on the left at the trimap border can be corrected by post-processing. The "bloom" from over-exposed hair on the right is conflated with defocus; the bright building panel in the background has no texture, and happens to align with the specular highlights from the hair; coherence terms in the regularization mislead the optimizer into classifying the panel as foreground and create an artifact too large to correct.*

similar but independent artifact occurs at sharp α edges when the magnitude of the α coherence regularization term is too large. Figure 13 shows a closeup of the matte pulled for the actor's shoulder from Figure 1. The subimage on the left has a large magnitude ($\beta = 0.3$) α coherence term, which leads to both excessive blurring and ringing. The ringing is a checkerboard because horizontal and vertical derivatives are handled separately. The subimage on the right shows the correct matte pulled with a reasonable ($\beta = 0.05$) coherence magnitude.

Figure 16 shows three frames of video on the left and their automatically extracted trimaps on the center. Because there is less information available at the edges of the image the algorithm tends to conservatively classify border pixels as unknown. Defocus is generally unreliable at the edge of an image, so we crop final results by the larger PSF radius. The final, post-processed matte is shown on the right.

As one would expect, the trimap estimation fails as the foreground object moves beyond the depth of field, and then the optimizer cannot proceed. Given a hand-painted trimap, the optimizer degrades more gracefully and tends to estimate a blurry foreground object and matte. At some point the defocus is too severe and the foreground object becomes classified as part of the background– which is desirable, since we define background as "beyond the depth of field."

9 Other Applications

9.1 Artificial Depth of Field

We can matte a subject back onto the reconstructed background, but choose the point spread functions and transformations arbitrarily. This allows us to render images with virtual depth of field, and even slight translation and zoom. Post-processed blur produces incorrect values at mixture pixels but still appears reasonable, as shown by Potmesil and Chakravarty [1983].

9.2 Video Filtering

Defocus is not the only effect we can apply when recompositing against the original background– any filter can be used to process the foreground and background separately using the matte as a selection region... e.g., hue adjustment, painterly rendering, motion blur (or deblur!), etc. Figure 1 (far right) shows an example of background desaturation to emphasize the foreground.

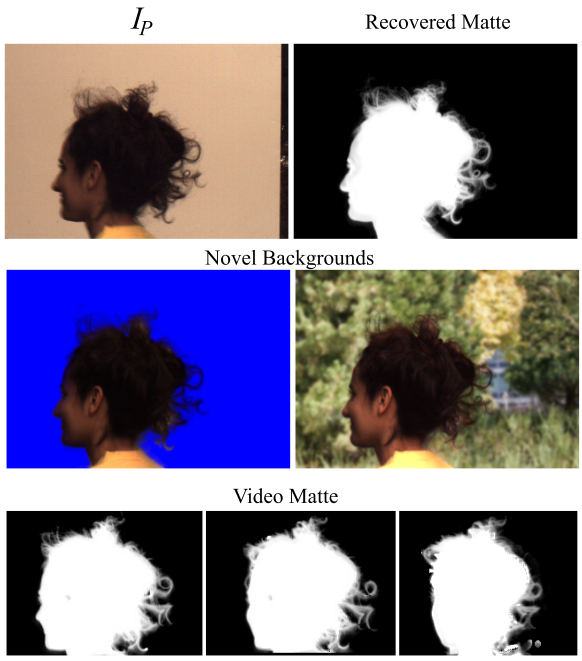

Figure 15: *Top: Another hair-extraction example, from real data. Pinhole, extracted alpha, extracted foreground, and composition over novel backgrounds. Note motion blur near the nose has been detected in the alpha channel. Bottom: Three frames of alpha from the video of the moving dark hair.*

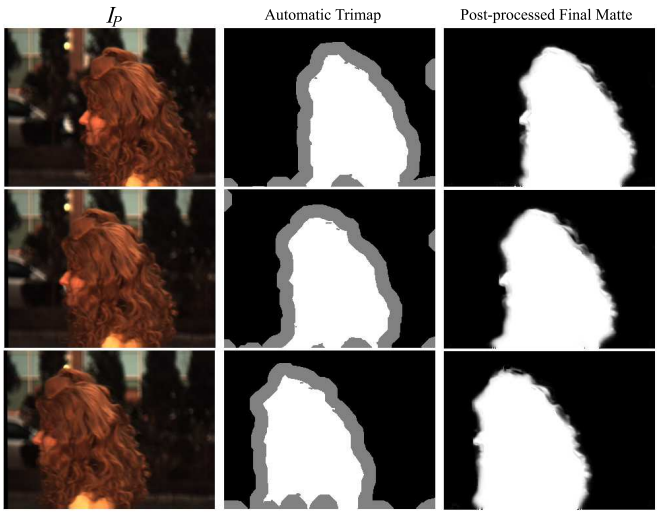

Figure 16: *Automatic trimap-extraction in a video sequence. Pinhole images at left, trimaps center, mattes at right.*

10 Limitations and Future Work

The sensors behind two beam splitters receive only 25% of the incident light. Thus our system requires stronger illumination than does a normal camera. Mitigating this, we use comparatively wide apertures to achieve narrow depth of field.

Just as green-screen matting limits actors to not wear green clothes or have green eyes, defocus matting has its own limitations. The most fundamental is that we require a depth discontinuity. Thus, for example, feet on a floor must be pulled using a different method. Under- and over-exposed areas and objects moving fast enough to produce significantly different motion blur for the differing exposures create artifacts that tend to push background objects into the foreground.

Like other natural image matting methods, our approach is limited to scenes where the foreground and background are visually distinguishable. If the boundaries of an actor, say, in a white shirt against a white wall, are hard to distinguish visually in an image, no natural image matting technique can succeed. Adding wavelength samples outside the visible spectrum (e.g., near field infra-red) increases the probability that similarly- colored materials can be distinguished. Investigating such a configuration is interesting future work, as is further experimentation with other optimization methods for solving the matting problem. Because only the Jacobian expressions are based on our choice of camera parameters, the defocus matting framework can be naturally extended to incorporate other camera parameters like variable focal length, high dynamic range, and stereo video.

Our gradient descent framework is in Matlab and we evaluate the error function in C on the CPU, but the initial image registration occurs on the GPU. The operations performed by the optimizer– convolutions with small filters, sparse matrix multiplication, and vector addition– are natural to also port to a graphics processor. Yet today's GPUs provide fewer than 32 bits of floating point precision and lack floating point blending (accumulation) support. We suggest that 64-bit precision with blending will enable a large class of applications to execute on GPUs, including matting.

We see promise in two are of future work. The first is the calibration problem. Although large objects still matte when the images are slightly miscalibrated, fine details are often lost. On ideal images, we recover high quality results. Thus we believe that a method for manufacturing a precisely aligned multiparameter camera, or one for efficiently producing perfect calibration on an imperfect camera would lead to production quality with the existing algorithm for high-frequency scenes.

For scenes with many low frequency areas, temporal and spatial coherence methods are likely to produce good results and are the second area of future work. We have briefly explored temporal coherence by using the previous frame's matte as an initial estimate for a new frame. However, for fast-moving objects it sometimes takes the optimizer longer to recover from motion error than starting with a fresh estimate. We also use temporal coherence in estimating the occluded background occluded areas by examining adjacent frames. Even when the background contains dynamic elements this improves the initial estimate substantially.

11 Acknowledgements

William Yerazunis and John Barnwell of MERL worked with us to design and manufacture the camera. We thank Shree Nayar (Columbia), Doug Roble (Digital Domain), Bill Freeman (MIT), and Michael Black (Brown) for their advice, and NVIDIA Corporation for Morgan's fellowship. The novel backgrounds are photographs by Shriram Krishnamurthi.

References

APOSTOLOFF, N. E., AND FITZGIBBON, A. W. 2004. Bayesian video matting using learnt image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 407–414.

ASADA, N., FUJIWARA, H., AND MATSUYAMA, T. 1998. Seeing behind the scene: analysis of photometric properties of occluding edges by the reversed projection blurring model. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 2, 155–67.

BEN-EZRA, M., AND NAYAR, S. 2004. Jitter camera: High resolution video from a low resolution detector. In *IEEE CVPR*, 135–142.

BHASIN, S. S., AND CHAUDHURI, S. 2001. Depth from defocus in presence of partial self occlusion. *Proceedingsof the International Conference on Computer Vision 1*, 2, 488–93.

BLAKE, A., ROTHER, C., BROWN, M., PEREZ, P., AND TORR, P. 2004. Interactive image segmentation using an adaptive gmmrf model. *Proceedings of the European Conference on Computer Vision (ECCV)*.

CHUANG, Y.-Y., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2001. A bayesian approach to digital matting. In *Proceedings of IEEE CVPR 2001*, IEEE Computer Society, vol. 2, 264–271.

CHUANG, Y.-Y., AGARWALA, A., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2002. Video matting of complex scenes. *ACM Trans. on Graphics 21*, 3 (July), 243–248.

DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 369–378.

FAVARO, P., AND SOATTO, S. 2003. Seeing beyond occlusions (and other marvels of a finite lens aperture). In *IEEE CVPR*, 579–586.

FLEISCHER, M., 1917. Method of producing moving picture cartoons. US Patent no. 1,242,674.

GLASSNER, A. S. 1995. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers, Inc.

HARALICK, R. M., STERNBERG, S. R., AND ZHUANG, X. 1987. Image analysis using mathematical morphology. *IEEE PAMI 9*, 4, 532–550.

HECHT, E. 1998. *Optics Third Edition*. Addison Wesley Longman, Inc.

HILLMAN, P., HANNAH, J., AND RENSHAW, D. 2001. Alpha channel estimation in high resolution images and image sequences. In *Proceedings of IEEE CVPR 2001*, IEEE Computer Society, vol. 1, 1063–1068.

LEVOY, M., CHEN, B., VAISH, V., HOROWITZ, M., MCDOWALL, I., AND BOLAS, M. 2004. Synthetic aperture confocal imaging. *ACM Trans. Graph. 23*, 3, 825–834.

MALVAR, H. S., WEI HE, L., AND CUTLER, R. 2004. High-quality linear interpolation for demosaicing of bayer-patterned color images. *Proceedings of the IEEE International Conference on Speech, Acoustics, and Signal Processing*.

NAYAR, S. K., AND BRANZOI, V. 2003. Adaptive dynamic range imaging: Optical control of pixel exposures over space and time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1168–1175.

NAYAR, S. K., WATANABE, M., AND NOGUCHI, M. 1996. Real-time focus range sensor. *IEEE PAMI 18*, 12, 1186–1198.

NOCEDAL, J., AND WRIGHT, S. J. 1999. *Numerical Optimization*. Springer Verlag.

PENTLAND, A. P. 1987. A new sense for depth of field. *IEEE PAMI 9*, 4, 523–531.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM Press, 253–259.

POTMESIL, M., AND CHAKRAVARTY, I. 1983. Modeling motion blur in computer-generated images. *Computer Graphics 17*, 3 (July), 389–399.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. on Graphics 23*, 3, 309–314.

RUZON, M. A., AND TOMASI, C. 2000. Alpha estimation in natural images. In *CVPR 2000*, vol. 1, 18–25.

SCHECHNER, Y. Y., KIRYATI, N., AND BASRI, R. 2000. Separation of transparent layers using focus. *International Journal of Computer Vision*, 25–39.

SMITH, A. R., AND BLINN, J. F. 1996. Blue screen matting. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, 259–268.

SUN, J., JIA, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Poisson matting. *ACM Transactions on Graphics* (August), 315 – 321.

YAHAV, G., AND IDDAN, G. 2002. 3dv systems' zcam. *Broadcast Engineering*.

ZITNICK, C. L., KANG, S. B., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2004. High-quality video view interpolation using a layered representation. *ACM Trans. on Graphics 23*, 3, 600–608.